

Программирование Python

Обработка текстов и работа с текстовыми форматами
данных
Черновик

Кафедра ИВТ и ПМ
ЗабГУ

2018

План

Регулярные выражения. Введение.

Языки разметки

XML

HTML

JSON

docx, odf, xlsx, ...

Обработка текстов на естественном языке

Ссылки и литература

Outline

Регулярные выражения. Введение.

Языки разметки

XML

HTML

JSON

docx, odf, xlsx, ...

Обработка текстов на естественном языке

Ссылки и литература

Регулярные выражения

Во время изучения чего-то нового, я самозабвенно выдумываю невероятные ситуации, в которых это умение поможет мне спасти мир

О нет! Убийца должно быть последовал за ней в отпуск!



Но чтобы узнать где он, нам нужно прочесть 200 Мб писем в поисках чего-то похожего по формату с адресом!



Это безнадежно!

Всем расступиться



Я знаю регулярные выражения



Регулярные выражения

Регулярные выражения (regular expressions) — формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов (wildcard characters).

Регулярные выражения

Для поиска используется строка-образец (**pattern**, «шаблоном», «маской»), состоящая из символов и метасимволов и задающая правило поиска.

Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.

Регулярные выражения

Краткая информация о способах задания шаблона приведена в справке по модулю.

```
help( re )
```

Регулярные выражение

Маской может выступать обычная подстрока. В этом случае поиск по регулярному выражению ничем не отличается от обычного поиска подстроки.

Однако вся мощь регулярных выражений кроется в использовании **метасимволов** - специальных символов и их комбинации.

Регулярные выражения

- ▶ `\w` - цифры, символы верхнего, нижнего регистра и `"_"`
- ▶ `\s` - пробел
- ▶ `\d` - число
- ▶ `\c` - управляющий символ

Аналогично можно ограничивать поиск символами, которые НЕ должны встретиться

- ▶ `\W` - не цифры, не символы верхнего, нижнего регистра и не `"_"`
- ▶ `\S` - не пробел
- ▶ `\D` - не число
- ▶ `\C` - не управляющий символ

Регулярные выражения

Если требуется задать собственный класс символов то можно использовать перечисления.

Перечисления задаются в квадратных скобках. Одно перечисление в фигурных скобках задаёт все возможные значения для *одного* символа.

Любая нечётная цифра

[13579]

Первые три буквы русского алфавита, либо заглавные либо строчные.

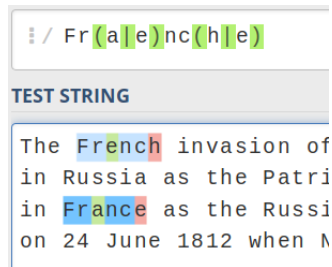
[а-вА-В]

Регулярные выражения

Если требуется найти найти похожие слова, отличающиеся одной или несколькими буквами можно задать шаблон таким образом:

`Fr(a|e)nc(h|e)`

Здесь запись `(a|e)` обозначает один символ, который может быть либо `a`, либо `e`



Регулярные выражения

Кванторы задают число повторений заданного символа или их группы

- ▶ Ровно три цифры
`\d{3}`
- ▶ Слово длиной от 4 до 6 букв в нижнем регистре
`[a-z]{4,6}`
- ▶ Слово в нижнем регистре из трёх букв или больше
`[a-z]{3,}`
- ▶ 0 или 1 перечисленных символов
`[qwerty]?`
- ▶ 0 или сколько угодно перечисленных символов
`[qwerty]*`

Последние выражения не применяются отдельно, так как им может соответствовать пустая строка.

Регулярные выражения

```
import re

text = "The French invasion of Russia, known \
in Russia as the Patriotic War of 1812 and \
in France as the Russian Campaign, began \
on 24 June 1812 when Napoleon's Grande Armée \
crossed the Neman Rive"

res = re.search("\d\d\d\d", text)
if res:
    print("Найденная подстрока: " + res.group(0))
    print("Позиция подстроки: " + str(res.span()))
```

Найденная подстрока: 1812

Позиция подстроки: (71, 75)

Регулярные выражения в текстовых редакторах

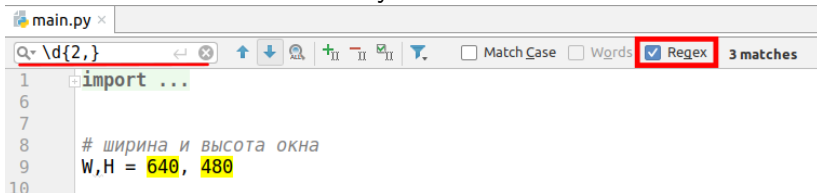
Регулярные выражения используются для гибкого поиска в больших текстах.

Например в программах Sublime text, Atom и PyCharm можно включить поддержку регулярных выражений в обычном поиске.

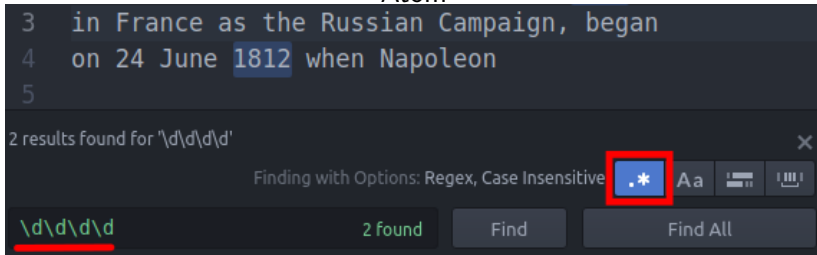
В Linux для поиска по регулярному выражению используется утилиты `grep` или `egrep`.

Регулярные выражения в текстовых редакторах

PyCharm



Atom



Примеры

`8(\s|-)\d\d\d(\s|-)\d\d\d(\s|-)\d\d\d\d`

Примеры

`8(\s|-)\d\d\d(\s|-)\d\d\d(\s|-)\d\d\d\d`

Поиск номера телефона вида 8-999-123-456. Вместо тире могут быть пробелы.

`[А-Я] . [А-Я] . [А-Я] [а-я]{2,}`

Примеры

`8(\s|-)\d\d\d(\s|-)\d\d\d(\s|-)\d\d\d\d`

Поиск номера телефона вида 8-999-123-456. Вместо тире могут быть пробелы.

`[А-Я] . [А-Я] . [А-Я] [а-я]{2,}`

Поиск строк вида "И. О. Фамилия"

Применение

Помимо поиска подстрок в текстах регулярные выражения применяются для замены подстрок, а также для проверки введённых пользователем данных.

Например с помощью регулярного выражения можно проверить введён ли корректный email-адрес, номер телефона или номер банковской карты.



The image shows a web form with a light gray background. At the top, there is a red rectangular box with the text "Error: Invalid email address:" in white. Below this, the form contains four labels on the left and their corresponding input fields on the right: "Name:" with a field containing "Mr John", "Password:" with a field containing ten dots, "Email:" with a field containing "mrjodh", and "Telephone:" with an empty field. At the bottom left of the form is a "Submit" button. In the bottom right corner of the slide, there are navigation icons and the text "17 / 37".

Error: Invalid email address:

Name:

Password:

Email:

Telephone:

Применение

Регулярные выражения учитывающие все аспекты искомой подстроки могут быть очень сложными и трудными в понимании. Поэтому следует применять и другие способы обработки текстовой информации, где регулярное выражение может использоваться только на одном их этапов.

Например можно ограничивать области текстов к которым применяются регулярные выражения если известна структура документа.

Если идёт речь о технических форматах документов, например html, то лучше использовать специальные методы разбора html документов.

Нечёткий поиск

```
import regex

# поиск подстроки amazing в строке amaging
# допустима одна или меньше опечатка ( e<=1 )
res = regex.match('(amazing){e<=1}', 'amaging')

if res:
    print("Найденная подстрока: " + res.group(0))
    print("Позиция подстроки: " + str(res.span()))
    print("Число опечаток, лишних вставок и недостающих символов: "
          + str(res.fuzzy_counts))
    print("Число позиций с опечатками, лишними вставками \
и недостающими символами: " + str(res.fuzzy_changes))
```

Найденная подстрока: amaging

Позиция подстроки: (0, 7)

Число опечаток, лишних вставок и недостающих символов: (1, 0, 0)

Число позиций с опечатками, лишними вставками и недостающими символами: ([3], [], [])

Нечёткий поиск

Три типа ошибок

- ▶ “i” - вставлен лишний символ
- ▶ “d” - нужный символ пропущен (удалён)
- ▶ “s” - символ замещён

“e” обозначает любую из этих ошибок

Нечёткий поиск

Примеры:

$i \leq 3$ permit at most 3 insertions, but no other types

$d \leq 3$ permit at most 3 deletions, but no other types

$s \leq 3$ permit at most 3 substitutions, but no other types

$i \leq 1, s \leq 2$ permit at most 1 insertion and at most 2 substitutions, but no deletions

$e \leq 3$ permit at most 3 errors

$1 \leq e \leq 3$ permit at least 1 and at most 3 errors

$i \leq 2, d \leq 2, e \leq 3$ permit at most 2 insertions, at most 2 deletions, at most 3 errors in total, but no substitutions

pypi.org/project/regex - regex docs

Outline

Регулярные выражения. Введение.

Языки разметки

XML

HTML

JSON

docx, odf, xlsx, ...

Обработка текстов на естественном языке

Ссылки и литература

Язык разметки (текста) — набор символов или последовательностей, вставляемых в текст для передачи информации о его выводе или строении. Примеры языков разметки HTML, XML, TEX

Outline

Регулярные выражения. Введение.

Языки разметки

XML

HTML

JSON

docx, odf, xlsx, ...

Обработка текстов на естественном языке

Ссылки и литература

XML

XML - (eXtensible Markup Language) — расширяемый язык разметки.

```
<?xml version="1.0" encoding="iso-8859-8" standalone="yes" ?>
<CURRENCIES>
  <LAST_UPDATE>2004-07-29</LAST_UPDATE>
  <CURRENCY>
    <NAME>dollar</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>USD</CURRENCYCODE>
    <COUNTRY>USA</COUNTRY>
    <RATE>4.527</RATE>
    <CHANGE>0.044</CHANGE>
  </CURRENCY>
  <CURRENCY>
    <NAME>euro</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>EUR</CURRENCYCODE>
    <COUNTRY>European Monetary Union</COUNTRY>
    <RATE>5.4417</RATE>
    <CHANGE>-0.013</CHANGE>
  </CURRENCY>
</CURRENCIES>
```

Outline

Регулярные выражения. Введение.

Языки разметки

XML

HTML

JSON

docx, odf, xlsx, ...

Обработка текстов на естественном языке

Ссылки и литература

HTML

HTML (HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык разметки документов во Всемирной паутине.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Example</title>
5          <link rel="stylesheet" href="sty
6      </head>
7      <body>
8          <h1>
9              <a href="/">Header</a>
10         </h1>
11         <nav>
12             <a href="one/">One</a>
13             <a href="two/">Two</a>
14             <a href="three/">Three</a>
15         </nav>
```

habr.com/post/280238 -Web Scraping с помощью python

Outline

Регулярные выражения. Введение.

Языки разметки

XML

HTML

JSON

docx, odf, xlsx, ...

Обработка текстов на естественном языке

Ссылки и литература

JSON

JSON (JavaScript Object Notation, обычно произносится как) — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

```
1  {  
2    "orderId": 12345,  
3    "shopperName": "Ivan Ivanov",  
4    "shopperEmail": "ivanov@example.com",  
5    "contents": [  
6      {  
7        "productId": 34,  
8        "productName": "Super product",  
9        "quantity": 1  
10     },  
11     {  
12       "productId": 56,  
13       "productName": "Wonderful product",  
14       "quantity": 3  
15     }  
16   ],  
17   "orderCompleted": true  
18 }
```

docs.python-guide.org/en/latest/scenarios/json/

Outline

Регулярные выражения. Введение.

Языки разметки

XML

HTML

JSON

docx, odf, xlsx, ...

Обработка текстов на естественном языке

Ссылки и литература

docx. Пример

```
from docx import Document

# открыть файл my_document.docx
doc = Document("my_document.docx")

# первый параграф
print( doc.paragraphs[0].text )

# вывод на экран всех параграфов документа
for par in doc.paragraphs:
    print(par.text)
```

python-docx.readthedocs.io/en/latest/user/quickstart.html

- ▶ <https://github.com/eea/odfpy>
- ▶ <https://openpyxl.readthedocs.io/en/stable/>

Outline

Регулярные выражения. Введение.

Языки разметки

XML

HTML

JSON

docx, odf, xlsx, ...

Обработка текстов на естественном языке

Ссылки и литература

Обработка текстов на естественном языке

- ▶ NLP - Natural Language Processing.
- ▶ Векторная модель слов
- ▶ pymorphy2.readthedocs.io/en/latest/user/guide.html - Морфологический анализатор pymorphy2
- ▶ Поиск имён и адресов в текстах: модуль Natasha

Outline

Регулярные выражения. Введение.

Языки разметки

XML

HTML

JSON

docx, odf, xlsx, ...

Обработка текстов на естественном языке

Ссылки и литература

Ссылки и литература

- ▶ regex101.com - online regex tester
- ▶ <http://www.exlab.net/files/tools/sheets/regexp/regexp.pdf> - шпаргалка по регулярным выражениям
- ▶ habr.com/post/115825 - Регулярные выражения, пособие для новичков.
- ▶ Регулярные выражения. Основы. Майкл Фицджеральд. 144с. 2015 г.

Ссылки и литература

Ссылка на слайды

github.com/VetrovSV/Programming